CSE1322L Assignment 7 - Fall 2024

Introduction:

It is not uncommon for live events to feature some sort of merchandise store, granting event attendees the opportunity to get themselves some memorabilia from the event, and the event organizers an additional source of revenue. As goods are sold, the event organizers will want to keep track of not only how many items were sold, but also how much they were sold for.

write a very simplified version of this system. All of our merchandise will be stored in a centralized location so we can more easily manage it (both physically and logically). However, we will have 3 different store fronts to keep the lines manageable. The idea is that whenever a store wishes to make a sale, it will pull out the first item available in the store and sell it to the customer. The stores will then update the event s ledger, keeping track of the revenue being generated and number of items sold.

This assignment makes use of Threads for parallelism and Stacks for data storage. Make sure to check the slides and videos on how those work.

Requirements

The features described below must be in your program.

A total of 4 classes: Store, Merchandise, MerchandiseStorage, and the Driver.

The Merchandise class has two public fields:

- A string named merchandise
- A Merchandise named next

  Merchandise is supposed to work like a Node in a LinkedList. Check the course material to understand what this means.

Merchandise only has an overloaded constructor which takes in a string, assigning it to the appropriate field, and sets next to null.

The MerchandiseStorage class has a single field called top , which is of type Merchandise.

- This class should behave like a stack, with the top field being treated as the top of a stack.

MerchandiseStorage has the following methods:

- addMerchandise(): takes in a string and returns nothing. This method behaves like the push() method of a stack: It must create a Merchandise object with the string in the argument, link it to the current top, and then make the newly created object the top
- retrieveMerchandise(): takes in nothing and returns a string. This method behaves like the pop() method of a stack, returning the current top s string after the current top has been replaced by its next.

If MerchandiseStorage is empty of Merchandise, it should return an empty string

The Store class has the following fields:

- A MerchandiseStorage field called pile
- A static integer called totalRevenue , initialized at 0
- A static integer called itemsSold , initialized at 0
- A static integer called nextId , initialized at 1
- An integer field called id

Store has only an overloaded constructor, which takes in a MerchandiseStorage, assigning it to the appropriate field. It then assigns nextId to id and increments nextId by 1.

Store has a getter for totalRevenue and itemsSold.

Store has a method called run(), which takes in no arguments and returns nothing

- **JAVA ONLY: You must make Store a subclass of Thread for run() to be an override**
- run() retrieves a merchandise from pile
- If the retrieved merchandise is an empty string, print Store {id} is done selling and then terminate the method
- Otherwise, increase itemsSold by 1 and increment totalRevenue as follows:
  keychain : +5
  t-shirt : +30
  plush : +50
- run() must keep retrieving merchandise from pile until it retrieves an empty string.

In your Driver, do the following:

- Create a MerchandiseStorage object
- Prompt the user for the number of keychains to add
- In a loop, call addMerchandise() with keychain as an argument however many times the user entered above
  For example, if the user enters they want to sell 50 keychains, you must call addMerchandise() with keychain as the argument 50 times.
- Prompt the user for the number of t-shirts to add
- In a loop, call addMerchandise() with t-shirt however many times the user entered above
- Prompt the user for the number of plushies to add
- In a loop, call addMerchandise() with plush however many times the user entered above
- Create three threads, one for each Store.
- Start all three threads
  Only start the threads AFTER you

Total revenue: ${Store.totalRevenue}
Number of items sold: {Store.itemsSold}
The show was a success!

C

Java Threads:

Defining Thread objects:

C# Threads:

Defining Thread behavior: You will create a method called run(). The code which you wish to be run in parallel with other threads must be inside this method. As such, most of your Store logic will be inside of run(). Keep in mind that, in theory, you can call this method anything you like, but the rubric will be specifically be looking for a method called run(). Make sure all of your parallel processing logic is in it.

Creating Thread objects: To create a thread object, you must pass to its constructor the name